Relational Algebra Continued

Where we are

- Last time: Relational algebra – formal definitions and examples of operators

$$\sigma, \pi, \times, \bowtie, \cap, \cup, -$$

- Today:
  - More advanced RA examples
  - Relational algebra equivalences and how to prove them
  - Bag relational algebra

Harder RA queries

- Find sailors who have reserved all boats
  - Remember the double "NOT EXISTS" in SQL?
- A two-step strategy:
  - think of a "magical operator" that would help us
  - figure out how to implement the "magical operator" using $\sigma, \pi, \times, \bowtie, \cap, \cup, -$

Sailors and Boats

sid	bid
s1	b1
s1	b2
s1	b3
s1	b4
s2	b1
s2	b2
s3	b2
s4	b2
s4	b4

R

bid
b1
b2
b3
b4

B

sid
s1
s2
s3
s4

S

sid
s1

Property of answer relation

- Answer is a subset of Sailors
- Answer $\times B \subseteq R$

- No larger subset of S has the above property

Division

- Integer division: 16/3 is the largest integer z such that z * 3 <= 16
- Relational division: R/B is the largest relation T such that

$$T \times B \subseteq R$$

Division, Sailors and Boats

sid	bid
s1	b1
s1	b2
s1	b3
s1	b4
s2	b1
s2	b2
s3	b2
s4	b2
s4	b4

R

bid
b2

B1

bid
b2
b4

B2

bid
b1
b2
b4

B3

sid
s1
s2
s3
s4

R/B1

sid
s1
s4

R/B2

sid
s1

R/B3

Division – formal definition

$$R/B = \{t \mid \forall b \in B \ (t \cdot b) \in R\}$$

Find the names of sailors who've reserved all boats

$$\rho(Tempsids, (\pi_{sid,bid}Reserves)/(\pi_{bid}Boats))$$

$$\pi_{sname}(Tempsids \bowtie Sailors)$$

Expressing R/B Using Basic Operators

- Now, need to express division using basic operators
- *Idea*: For *R/B*, compute all values that are not \`disqualified'.
  - *t* value is *disqualified* if by attaching *b* value from *B*, we obtain a *tb* tuple that is not in *R*.
  - Let A be all the attributes that are in R but not in B

Disqualified *r* values: $\pi_A((\pi_A(R) \times B) - R)$

R/B: $\pi_A(R) -$ all disqualified tuples

More extensions

- In classical RA, no support for aggregation (MIN/MAX/COUNT) or GROUP BY
- But there are extensions that introduce such operators
  - Beyond the scope of this class

Relational Algebra Equivalences

- *Selections:* $\sigma_{c1 \wedge \ldots \wedge cn}(R) \equiv \sigma_{c1}\left(\ldots\ \sigma_{cn}(R)\right)$ (Cascade)

  $\sigma_{c1}\left(\sigma_{c2}(R)\right) \equiv \sigma_{c2}\left(\sigma_{c1}(R)\right)$ (Commute)

- *Joins:* $R \bowtie (S \bowtie T) \equiv (R \bowtie S) \bowtie T$ (Associative)

  $(R \bowtie S) \equiv (S \bowtie R)$ (Commute)

More equivalences

- Do projections and selections commute?
- What about selection and join?
- Projection and join?
- See textbook for more equivalences (15.3)

Proving equivalences

- Because we have formally defined our RA operators, can PROVE that the equivalences are true.

Proving equivalences

- Some equivalences may seem obvious, but others do not
- Eg our two definitions of division

$$R/S = \{r \mid \forall s \in S \ (r \cdot s) \in R\}$$

$$\pi_A(R) - \pi_A((\pi_A(R) \times S) - R)$$

Proving equivalences

- Let's see how to prove a simple equivalence

$$\pi_A(R) \equiv \pi_A(\pi_B(R)),\ \ A \subseteq B$$

- Two queries are equivalent if return same set of tuples on every database
  - Otherwise, the queries have a different meaning!

Some NON-equivalent queries

```
SELECT DISTINCT S.sname
FROM  Sailors S, Boats B1, Boats B2, Reserves R1,
Reserves R2
WHERE  S.sid=R1.sid AND R1.bid=B1.bid
  AND  S.sid=R2.sid AND R2.bid=B2.bid
  AND (B1.color='red' AND B2.color='blue');
```

```
SELECT S.sname
FROM Sailors S, Boats B, Reserves R WHERE
R.sid=S.sid AND R.bid=B.bid
AND B.color='red'
INTERSECT
SELECT S.sname
FROM Sailors S, Boats B, Reserves R WHERE
R.sid=S.sid AND R.bid=B.bid
AND B.color='blue';
```

- These return a different answer on a DB with two sailors called Bob, one of whom reserves red boat and other reserves blue boat

Proving equivalences

- Let's see how to prove a simple equivalence

$$\pi_A(R) \equiv \pi_A(\pi_B(R)),\ \ A \subseteq B$$

- Two queries are <u>equivalent if return same set of tuples on every database</u>
  - Otherwise, the queries have a different meaning!
- So, need to prove on any DB the two sets above are equal.
  - How?

$$\pi_A(R) \equiv \pi_A(\pi_B(R)),\ \ A \subseteq B$$

- Show every element in LHS set is in RHS set
- And vice versa
- So need to show 2 things for arbitrary R:
- If $t \in \pi_A(R)$ then $t \in \pi_A(\pi_B(R))$
- If $t \in \pi_A(\pi_B(R))$ then $t \in \pi_A(R)$

$$\pi_A(R) \equiv \pi_A(\pi_B(R)), \; A \subseteq B$$

- If $t \in \pi_A(R)$ then $t = s[A]$ for some $s \in R$
  - No magic here, just applying def of projection!
- Now, consider tuple s as it is fed through our other query $\pi_A(\pi_B(R))$
  - Yields a result tuple $(s[B])[A]$ (again from def)
  - If we can show $(s[B])[A] = t$ we are done!
  - Because it will follow that $t \in \pi_A(\pi_B(R))$

$$\pi_A(R) \equiv \pi_A(\pi_B(R)),\ A \subseteq B$$

- Need to show $(s[B])[A] = t$
- But know $t = s[A]$
- So need to show $(s[B])[A] = s[A]$

- This is clear from the fact that $A \subseteq B$
- And from the definition of []

Reminder from last lecture

- For a tuple t, let $t[A]$ denote the restriction of tuple t to exactly the attributes in A
- Suppose $A = \{a_1, a_2, \cdots a_k\}$
- Suppose $t(a)$ denotes value of tuple t for attribute a
- Then $t[A] = t(a_1) \cdot t(a_2) \cdots t(a_k)$
- The dot represents concatenation

$$\pi_A(R) \equiv \pi_A(\pi_B(R)),\ \ A \subseteq B$$

- So far we have proved one direction (the "forward" direction) of this equivalence
- Other direction – pretty similar
- Start with tuple t in result of $\pi_A(\pi_B(R))$
- Show that it also appears in result of $\pi_A(R)$
  - Will need to use definition of projection
  - Will need to trace back the origin ("provenance") of t to some $s \in R$

Proving equivalences

- Remember to prove in both directions
- And make it clear which direction you are proving at all times
- Will be using definitions of the operators
- Will be tracing a tuple back through the operators to original relation
  - And then back down "on the other side"
- Some proofs will be more "exciting" than others

Bag relational algebra

- In the Relational Algebra we saw, all inputs and outputs to the operators are sets
  - No duplicates
- But in practice want to be able to reason about *bags*
  - A bag is a set that may contain repetitions/ multiplicities
    - {1, 1, 2, 3} is a bag but not a set
    - {1, 2, 3} is a set and also a bag

Why bags?

- Basic relations you use are still (most likely) sets
  - If they have a primary key they certainly are
- But it can be handy to reason about bags as intermediate steps in evaluation
  - The π operator removes duplicates
  - But this may be an expensive operation
    - So why do it if we don't care about unique values?
  - Would be handy to have a "projection that keeps duplicates" operator

Bag relational algebra

- Can define an algebra on relations that are bags

R

a	b
1	2
1	2
3	3

- Will have all the same operators, but need to explain how they work on bags

Selection

- Works the same way as before, it can just output a bag this time

R

a	b
1	2
1	2
3	3

$\sigma_{a=1}(R)$

a	b
1	2
1	2

Projection

- Same as before, just doesn't eliminate duplicates

R

a	b
1	2
1	2
3	3

$\pi_a(R)$

a
1
1
3

Cross product and join

- Same as before, treat any duplicates as though they were separate tuples

R

a	b
1	2
1	2
3	3

S

c	d
7	8
9	2

$R \times S$

a	b	c	d
1	2	7	8
1	2	9	2
1	2	7	8
1	2	9	2
3	3	7	8
3	3	9	2

Union, intersection, difference

- $R \cup_B S$
  - Each element appears as many times as it appears in R plus as many times as it appears in S
- $R \cap_B S$
  - Each element appears the minimum number of times it appears either in R or S
- $R -_B S$
  - Each element appears the number of times it appears in R, minus # times appears in S, but never <0 times.

Examples

R

a	b
1	2
1	2
3	4

S

a	b
1	2
3	4
3	4

a	b
1	2
1	2
1	2
3	4
3	4
3	4

$$R \cup_B S$$

a	b
1	2
3	4

$$R \cap_B S$$

a	b
1	2

$$R -_B S$$

Union, intersection, set difference

- If you want to play with those, try UNION ALL, INTERSECT ALL, EXCEPT ALL
- In Postgres, because MySQL doesn't support INTERSECT and EXCEPT…

- Basically: real SQL implements a combination of set and bag semantics

Equivalences

- Some classical equivalences no longer apply in bag relational algebra
  - E.g. $R \cup R = R$
- To prove an equivalence, now need to show:
  - For every tuple t, if appears with multiplicity k (i.e. k times) on RHS, appears k times on LHS
  - And vice versa

Clarification

- Bag relational algebra is useful in practice, but some courses (and your textbook) don't talk about it
- In the remainder of this course, when we ask you to write something in "relational algebra", assume we mean set RA
  - If we don't, we will mention the bag RA explicitly
- Practicum folks – you are working in the bag RA

Relational Algebra Summary

- Basic operators and how to use them
  - Selection, projection, cross product, union, intersection, difference
- Derived operators
  - E.g. join, division (defined in terms of basic ones)
- Equivalences and how to prove them
- Bag RA